Return To Agenda Page

ARC's SIMD Extensions for Multimedia Applications

Nigel Topham

Chief Architect

Quest for an Efficient High-Performance Embedded Solution

- Embedded media-rich applications need greater performance
- Cannot rely on increasing clock speed
- Power consumption is a critical constraint
- Parallelism is the future – but what form will it take?

~~VLIW~~

Why Not?
- Compiler complexity
- Issue bandwidth
- Die-area bloat
- Poor sustained performance
- High power consumption

~~Superscalar~~

Why Not?
- Hardware complexity
- Modest gains
- Not scalable
- May play a part, but is not itself a solution

~~Simultaneous Multithreading (SMT)~~

Why Not?
- Offers a throughput gain only
- May make sense for a large, inefficient core
- More effective to build multicore ARC systems

☑ Data-level Parallelism

Requirements
- ☑ SIMD, short vector
- ☑ Multiple instruction issue
- ☑ Configurable
- ☑ Application-specific
- ☑ High memory bandwidth

☑ Multiprocessor Parallelism

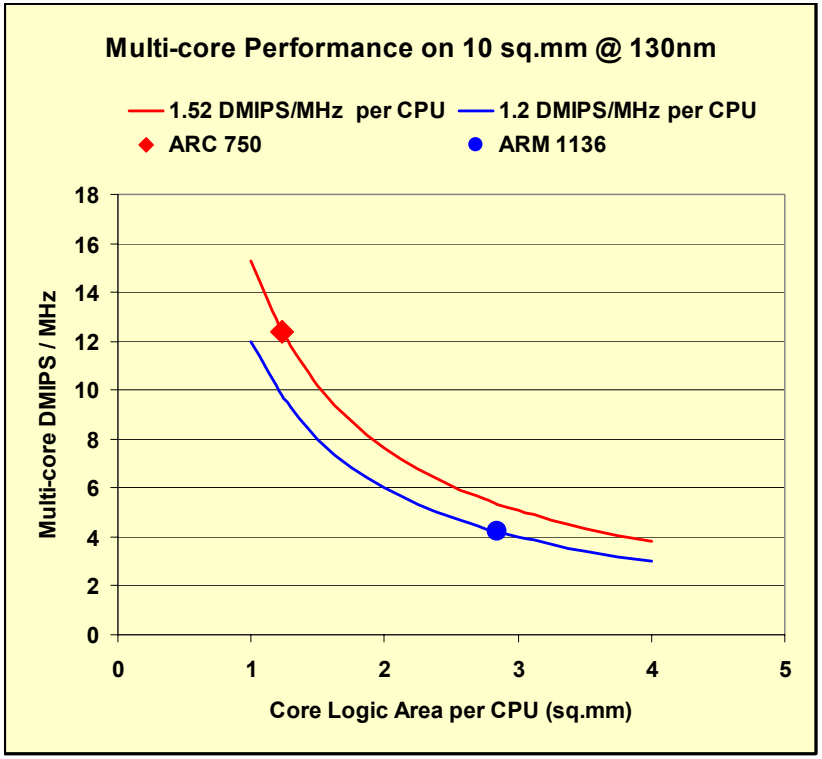
Requirements
- ☑ SMP
- ☑ Heterogeneous
- ☑ Configurable
- ☑ Application-specific
- ☑ Inter-processor communications

ARC International and Multicore

- ARC's multicore history
  - 40% of ARC-based deployments involve multiple cores
  - Average number of cores per design = 6
- Small, fast, low-power processing elements are key
- Latest ARC™ 750D core is ideal for multicore implementations

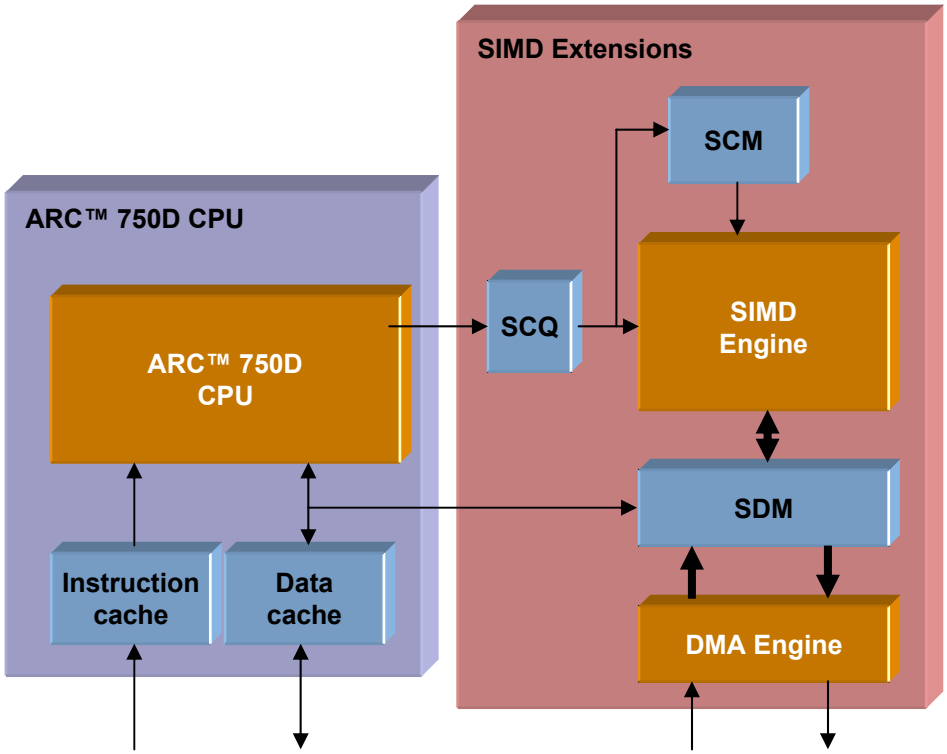
ARC™ 750D Performance		
	130 nm	90 nm
Frequency	533 MHz	700 MHz
DMIPS / MHz	1.526	1.526
DMIPS / CPU	813	1068

- ARC's multicore roadmap
  - Symmetric multiprocessor (SMP)
  - Heterogeneous scalar / SIMD



Introducing ARC's SIMD Extensions

- High-performance SIMD extensions extend ARC™ 750D core
- 128-bit data-parallel operations for high-throughput processing
- Decoupled-control concept yields low-cost superscalar performance
- Media-oriented dual-channel two-dimensional DMA engine
- All delivered as ARCompact™ compatible ISA extensions



SCQ – SIMD Code Queue

SCM – SIMD Code Memory

SDM – SIMD Data Memory

SIMD Programming Model

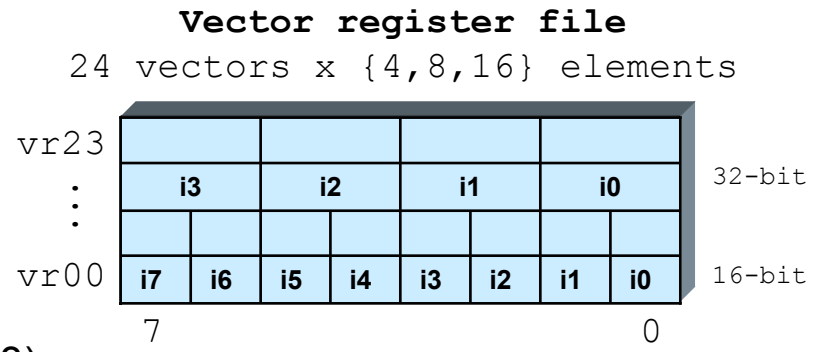
- Extends ARCompact™ ISA
- 128-bit vectors
  - 4 x 32 bits, 8 x 16 bits, 16 x 8 bits
  - Type depends on instruction
- Up to 64 vector registers
  - 24 in first implementation (vr00 – vr23)
- Scalar 16-bit registers (i0 – i7)
  - Eight scalar data or address values
  - Synonymous with vector register 0
- Vector accumulator
  - Third operand and result for some ops
  - 32-bit or extended 40-bit elements

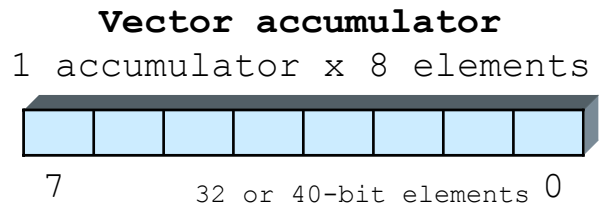
Broadcast multiply with accumulation

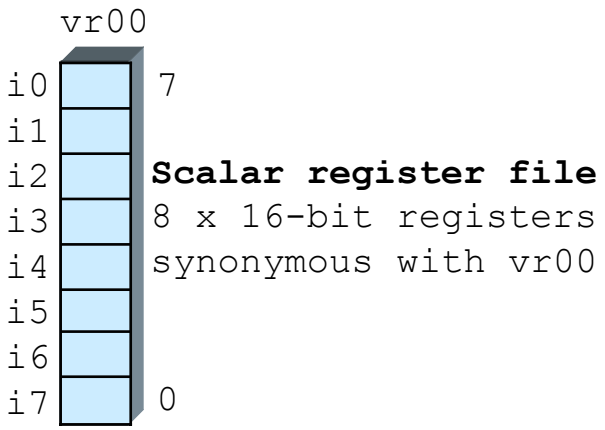
```
vbmulaw vr08, vr02, r3
```

Semantics

```
for (i = 0; i < 8; i++)
  vr08[i] = acc[i] = acc[i] + r3*vr02[i]
```

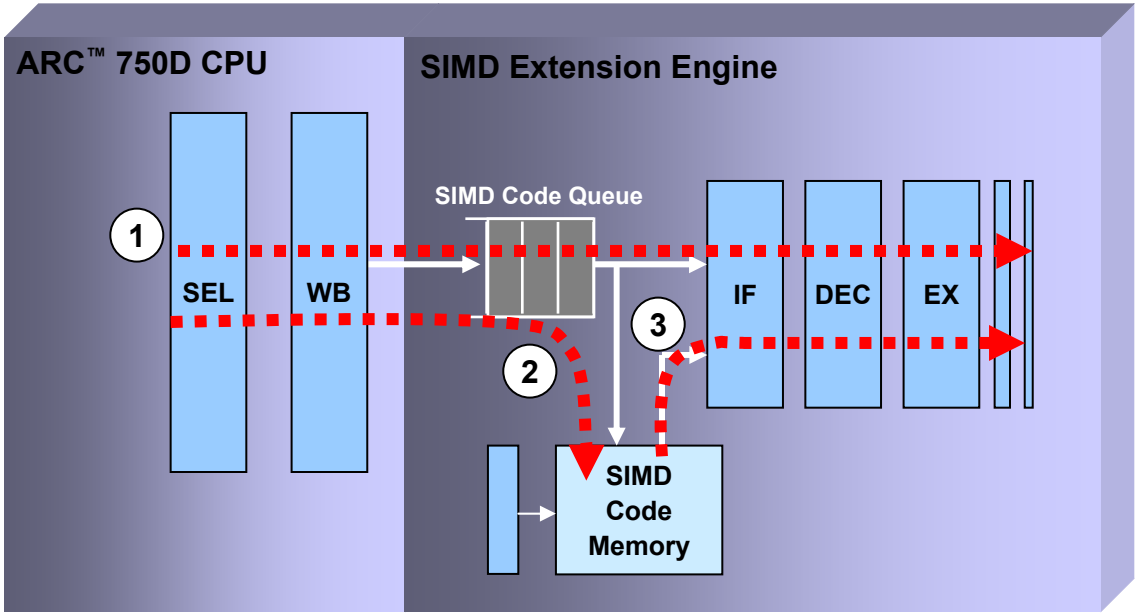






Decoupling Scalar and SIMD Execution

- Closely coupled and decoupled execution modes



1. Closely coupled execution — instructions issued directly from ARC™ 750D core
2. Macro record — instructions fetched by 750D core and stored in SCM
3. Decoupled execution:
   - 750D core issues VRUN with macro start address in SCM
   - SIMD engine issues instructions autonomously from SCM
   - Thereafter, 750D core and SIMD engine operate in parallel

Record / Replay Enables Parallel Execution

- Coupled model embeds SIMD code with ARC™ 750D core code
- VRUN initiates macro sequence
- Many VRUNs can be queued

Tightly coupled execution model

```
example_loop:
        push_s  r13
        mov_s   r13,r6
        vmovzw  vr00,r1,1
        vmovw   vr00,r2,2
        vmovw   vr00,r0,4
        vmovzw  vr23,r4,1
        vmovw   vr23,r5,2
        vmovw   vr23,r3,4
        cmp_s   r13,0
        mov.gt  lp_count,r13
        lpgt    .L01
        vld128  vr01,[i0]
        vld128  vr02,[i0,0x10]
        vmrb    vr01,vr02,vr01
        vld128  vr03,[i1]
        vavrb   vr01,vr01,vr03
        vst128  vr01,[i2]
        vaddw   vr00,vr00,vr23
.L01    j       r13
```

Decoupled execution model

```
example_loop:
        push_s  r13
        mov_s   r13,r6
        vmovzw  vr00,r1,1
        vmovw   vr00,r2,2
        vmovw   vr00,r0,4
        vmovzw  vr23,r4,1
        vmovw   vr23,r5,2
        vmovw   vr23,r3,4
        cmp_s   r13,0
        mov.gt  lp_count,r13
        lpgt    .L01
        vrun    r7
.L01    j       r13
```

Macro sequence pre-recorded in SCM

```
[r7]:   vld128  vr01,[i0]
        vld128  vr02,[i0,0x10]
        vmrb    vr01,vr02,vr01
        vld128  vr03,[i1]
        vavrb   vr01,vr01,vr03
        vst128  vr01,[i2]
        vaddw   vr00,vr00,vr23
```

SIMD Instruction-Set Overview

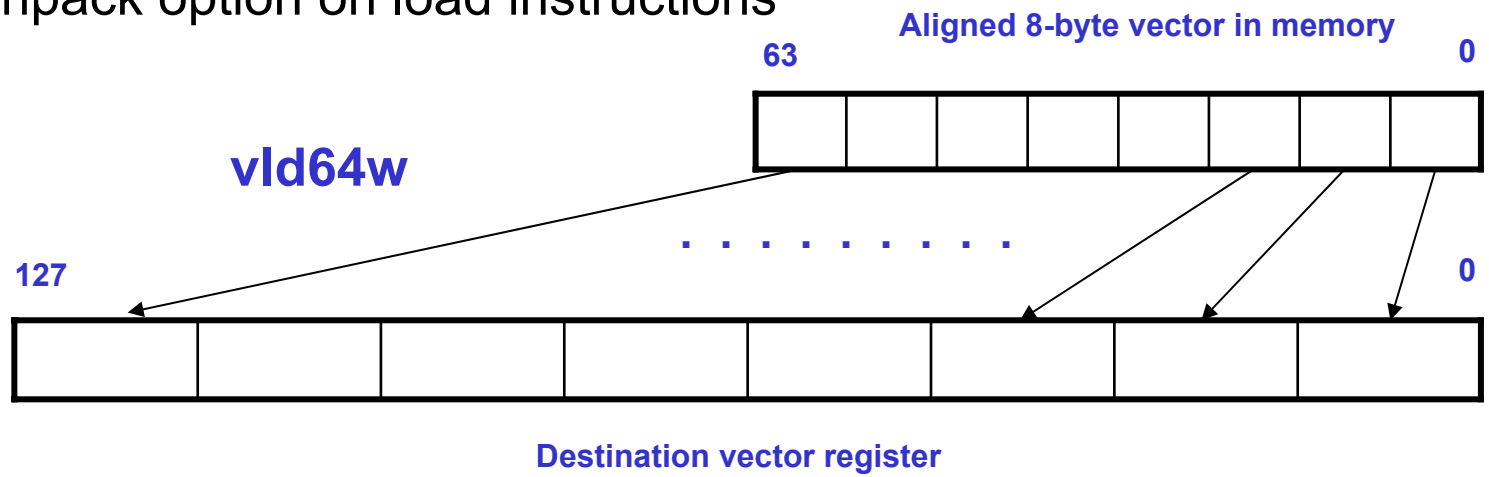
- 104 new vector instructions extend ARCompact™ ISA
- Carefully selected to optimize media application performance

Load	Store	Move		Logic		Arith			Compare
vld32	vst16	vmw	vmvaw	vand	vandaw	vaddw	vbaddw	vaddaw	veqw
vld64	vst32	vmvzw		vor		vsubw	vbsubw	vsubaw	vnew
vld64w	vst64	vmow	vmovaw	vxor	vxoraw		vbrsubw		vlew
vld32wl	vst128	vmovzw		vbic	vbicaw	vsummw			vltw
vld32wh	vst128r					vaddsuw			
vld128						vavb			
vld128r						vavrb			
						vmulw	vbmulw	vmulaw	
						vmulfw	vbmulfw	vmulfaw	
								vbmulaw	
						vmaxw	vbmaxw	vmaxaw	
						vminw	vbminw	vminaw	
						vabsw		vabsaw	
						vdifw		vdifaw	
						vsignw			

Permute	Align		Scale	Pack	Unpack	Misc	Special	Macros	DMA
vexch1	vmr1w	vmr1aw	vasrw	vasrpwb	vupbw	vnop	vd6tapf	vrec	vdirun
vexch2	vmr2w	vmr2aw	vasrrw	vasrrpwb	vupsbw	vint	vh264ft	vendrec	vdorun
vexch4	vmr3w	vmr3aw	vasrsrw		vupbaw		vh264f	vrun	vdiwr
	vmr4w	vmr4aw			vupsbaw		vvc1ft	vrecrun	vdowr
	vmr5w	vmr5aw					vvc1f		vdird
	vmr6w	vmr6aw							vdord
	vmr7w	vmr7aw							
	vsr8	vsr8aw							
	vmrb								

Data Movement

- Vector load or store up to 128 bits per cycle from SDM
- DMA moves data between SDM and external RAM
- Unpack option on load instructions

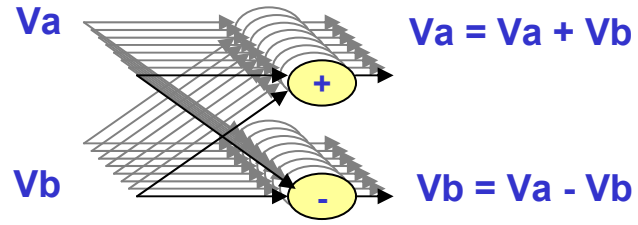


- Move vector elements (vector-to-vector move)
- Replicate scalar value (scalar-to-vector broadcast)
- Selective element assignment, determined by bit mask
- Deselected elements may be cleared or left intact

Vector Arithmetic

- Two-point vectorized butterfly – ideal for DCT and iDCT

vaddsuw Va, Vb

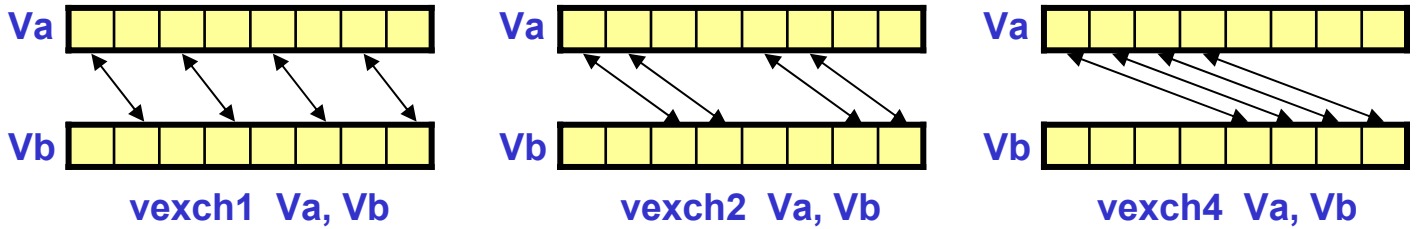


- Vector multiplication
  - Vector x vector, or vector x scalar
  - Eight 16 x 16 multipliers
  - 16-bit integer or 1Q15 fractional results with rounding

- Byte-vector averaging for motion compensation
  - Half- and quarter-pixel motion vectors require block interpolation
  - Capable of 16 byte-average calculations per cycle
  - Hence, *h* cycles to interpolate a 16 x *h* block

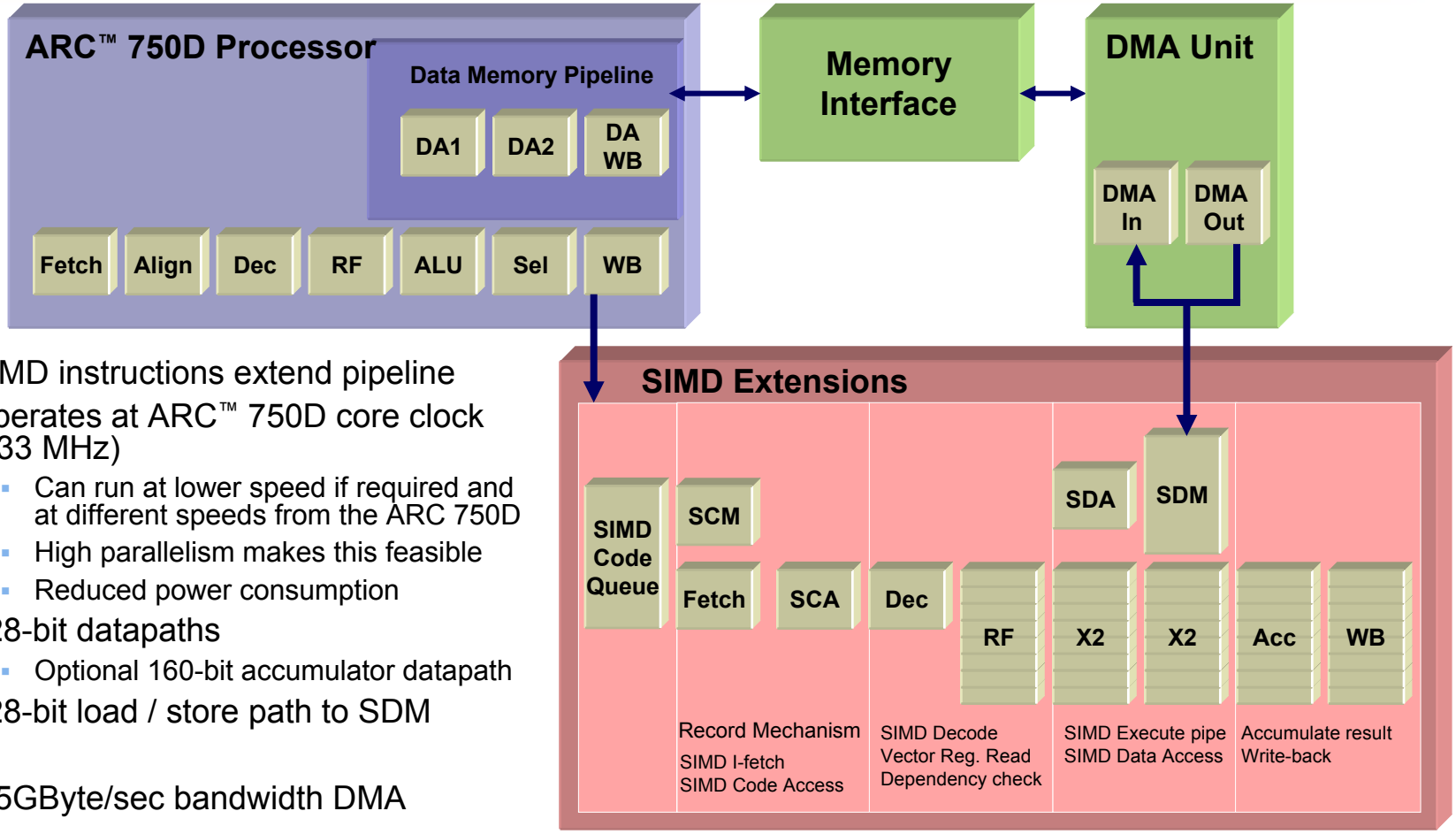
Permutations and Matrix Transposition

- 2D block transforms have horizontal and vertical passes
- Matrix transposition occurs between passes (4 x 4 or 8 x 8)
- Permutation instructions yield fast transposition



- Throughput:
  - One 4 x 4 matrix transposed in 4 instructions
  - Four 4 x 4 matrices transposed in 8 instructions
  - 8 x 8 matrix transposed in 12 instructions

SIMD Extensions Pipeline



- SIMD instructions extend pipeline
- Operates at ARC™ 750D core clock (533 MHz)
  - Can run at lower speed if required and at different speeds from the ARC 750D
  - High parallelism makes this feasible
  - Reduced power consumption
- 128-bit datapaths
  - Optional 160-bit accumulator datapath
- 128-bit load / store path to SDM
- 8.5GByte/sec bandwidth DMA
- Decoupled control
  - Independent SIMD I-fetch from SCM
  - Executes SIMD basic blocks with VRUN
  - 750D core queues future blocks in SCQ
  - 750D core “gets ahead” of SIMD engine

ARC™ Multimedia Subsystems Exploit SIMD Extensions

- Multistandard software-based platforms for audio/video media processing
- ARC™ Media Subsystem = ARC Video + ARC Sound Subsystem technologies
  - ARC Video: MPEG 1, 2, & 4, H.264, VC-1…and any future codecs
  - ARC Sound: All popular audio codecs
- Wide performance range
  - Low-frequency operation for mobile low-power applications
  - 600 core family sufficient for ARC Sound
  - SIMD extensions are powerful enough for H.264 and VC-1 decode at D1 resolution
- ARC Sound Advanced Subsystem
  - 12-channel AAC or MP3 encode, running on SIMD extensions

	Dedicated Hardware	General-Purpose CPU	ARC™ Media Subsystem
Flexibility	**None**	**Unlimited**	**Unlimited**
Die area	**Additive per codec**	**Fixed but high**	**Fixed and low**
Power consumption	**Low**	**High, due to high clock frequency**	**Low**
Operating frequency	**Very low**	**Very high**	**Low**

- The compelling case for software-based media processing

- Combining SIMD, multicore, and custom extensions is key

SIMD Algorithms: MPEG DCT and Inverse DCT

- DCT based on Arai, Agui, and Nakajima's scaled DCT[1]
  - Originally 8-point 1-D transform required 5 multiplies and 29 adds
  - ARC™ Multimedia subsystem implementations exploit vector butterfly extensively
  - Computes 8 x 8 1-D transform in 24 instructions
  - Full 2-D 8 x 8 DCT requires only 88 instructions
  - Quantization requires a further 8 instructions
  - 9.4x speedup due to SIMD extensions, compared with basic 750D core
- iDCT uses modified version of Loeffler's algorithm[2]
  - Originally 8-point 1-D transform required 11 multiplies and 29 adds
  - ARC Multimedia subsystems use 16 multiplies and 16 other ops (add, sub, butterfly)
  - Computes 8 x 8 1-D transform in 32 instructions
  - Full 2-D 8 x 8 IDCT requires only 116 instructions
  - 10.4x speedup due to SIMD extensions, compared with basic 750D core
  - Easily satisfies IEEE 1180 accuracy test

1. Arai, Agui, and Nakajima, Trans. IEICE E-71 (11): 1095
2. Loeffler, Ligtenberg, and Moschytz, Proc. ICASSP '89, pp. 988-991

Taking Care of Details: Numerical Fidelity

- Typically, integer or fractional de-scaling is via arithmetic right shift
- However, accurate rounding is critical to numerical fidelity
- No rounding
  - Compute as:

    (x >> d)
  - Mean error = $-0.5 + 0.5 * 2^{-d}$
- Simple rounding
  - Compute as:

    (x + (1 << (d-1))) >> d
  - Mean error = $0.5 * 2^{-d}$
- Symmetric rounding in ARC™ SIMD extensions
  - Compute as:

    (x < 0) ? (x + (1 << (d-1)) – 1) >> d : (x + (1 << (d-1))) >> d
  - Mean error is ZERO
  - Implemented in ARC Media subsystems right shift
  - Enables 16-bit fast DCT / iDCT to meet IEEE 1180 compliance

Custom SIMD Extensions: H.264 and VC-1

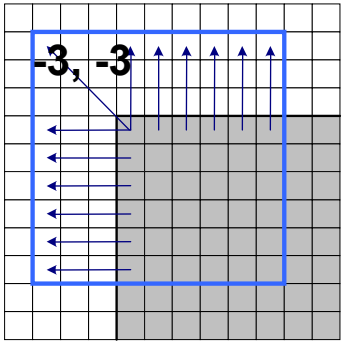
- SIMD ISA extended to support H.264 and VC-1
- Complex filtering operations split across two instructions
  1. Test deblocking conditions and set deblocking values
  2. Apply deblocking values to block boundary pixels
- H.264 and VC1 deblocking filters
  - vh264ft, vvc1ft – horizontal filter test
  - vh264f vvc1f – horizontal filter calculation
- Key optimization for H.264 and VC1 software implementation
  - Hides control complexity
  - Exposes irregular pixel-level parallelism to hardware
  - Speedup of 15x – 25x, exceeding the 8x expected of 8-way SIMD

Custom Scalar Extensions: Entropy Decoding

- Media codecs each define a bit-stream syntax
- Symbols encoded using a range of techniques
  - Huffman coding (JPEG, MPEG, VC-1)
  - Arithmetic coding (H.264)
  - Most complex schemes are context adaptive
- Decoding is a serial process – cannot be parallelized
- Goal is to decode one symbol per clock period (or instruction)
- ARC's scalar extension instructions provide ideal mechanism
- ARC solution:
  - Implement a single-cycle hardware decoder via extension instructions
  - All relevant codecs supported directly
    - VC-1, MPEG-1, 2, and 4 VLC and RVLC tables
    - H.264: CAVLC, CABAC, Exp-Golomb algorithms
- Major benefits:
  - No software VLC tables – no D-cache pollution from sparse structures
  - No D-cache dependency – predictable decode delays
  - Low die area cost
  - Extremely fast

Media-Aware Block DMA

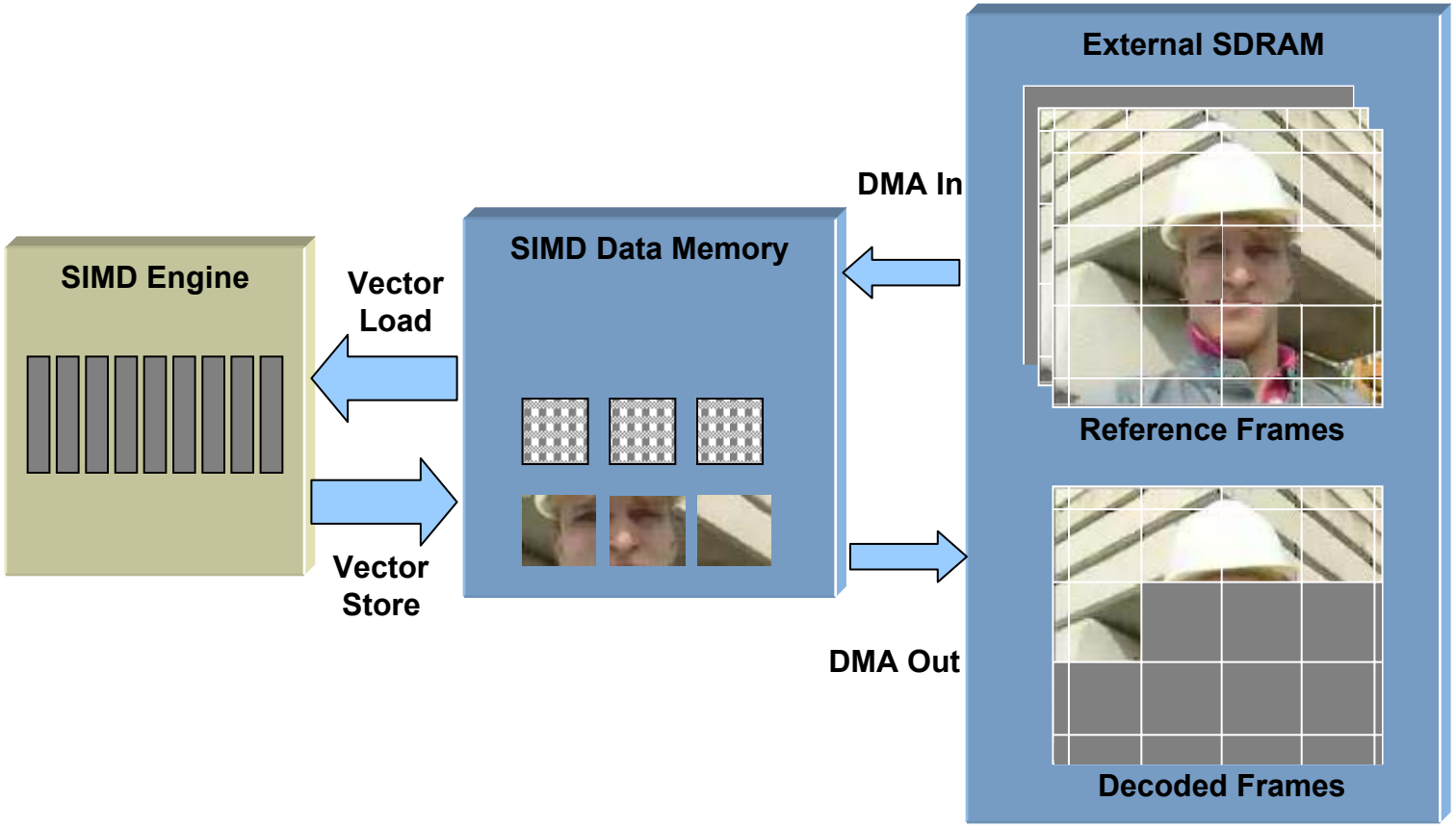
- Lightweight command interface – fast DMA initiation
- Frame descriptors held in SDM
- DMA commands specify frame and (x,y) location in frame
- Clipping and filling
  - Blocks can extend outside frame
  - DMA fills or clips appropriately
  - A feature of H.264 and VC-1



- DMA is SDRAM aware
  - Address reordering to minimize page-boundary crossing
- Supports both interlaced and progressive scan

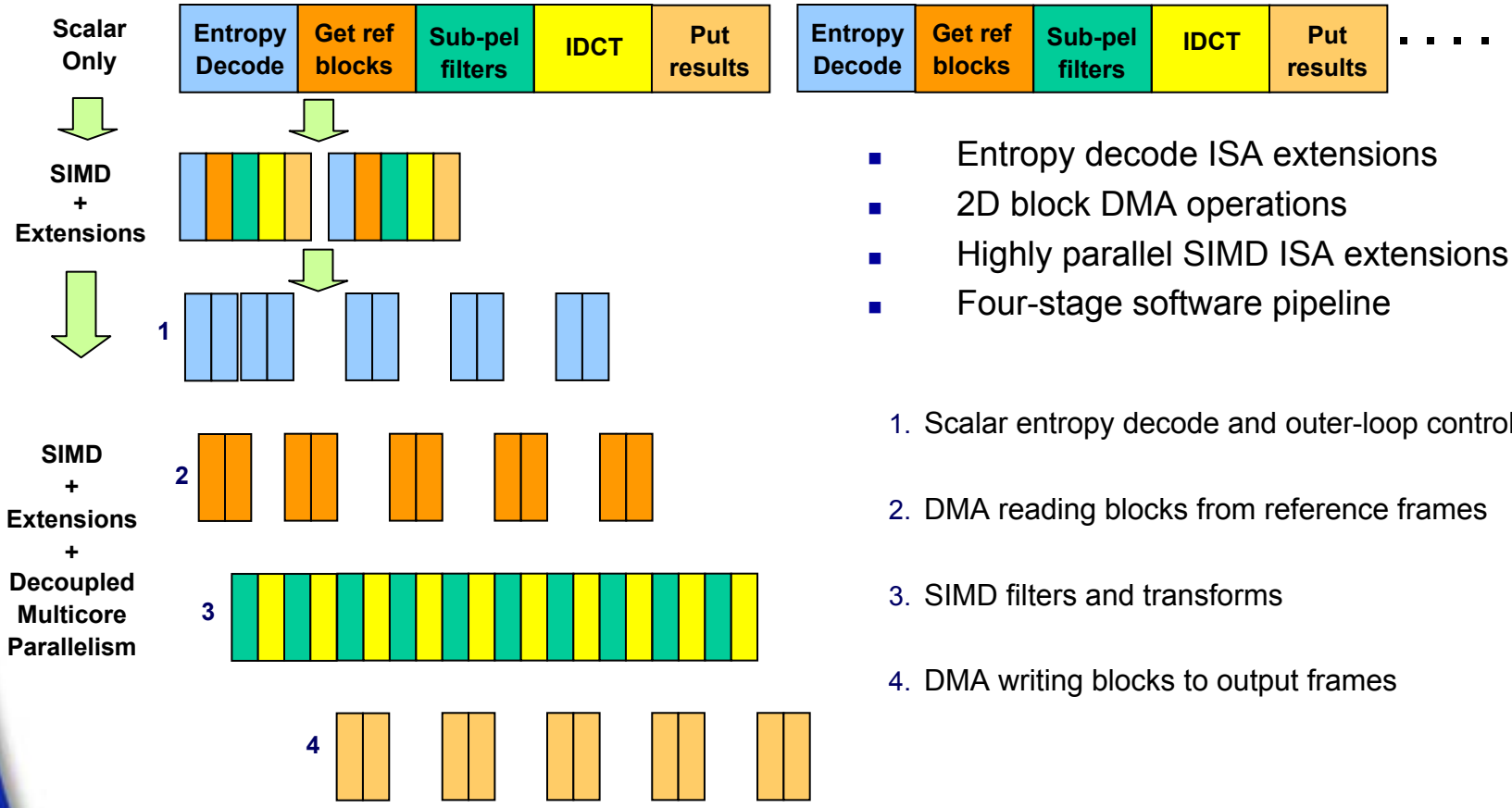
Scalar, SIMD, and DMA = 3-Way Parallelism

- Vector loads and stores address SDM (fast, on-chip RAM)
- Reference pictures typically stored in external SDRAM
- DMA moves 2D blocks between SDM and external SDRAM



Extensions + Parallelism = Multimedia Solution

- Combines parallel processing with ISA extensions
  - Parallelism among three major units
  - ISA extensions for entropy decode and vector processing



- Entropy decode ISA extensions
- 2D block DMA operations
- Highly parallel SIMD ISA extensions
- Four-stage software pipeline

1. Scalar entropy decode and outer-loop control
2. DMA reading blocks from reference frames
3. SIMD filters and transforms
4. DMA writing blocks to output frames

SIMD Video Decode Performance

- Exceptional performance for MPEG, H.264, and VC-1
- High sustained levels of data parallelism
- Illustrated with selection of key video computations
  - Coded and fully functional on ARCangel™ 4 development platform
  - Cycle times include block load / store as well as SIMD computation

Codec	Computation	SIMD clocks	Block	Clocks / pixel-op	SIMD Speedup
MPEG-1 MPEG-2 MPEG-4	Forward DCT 8x8	113	8 x 8	1.77	**9.4**
	Inverse DCT 8x8	129	8 x 8	2.02	**10.4**
	Matrix transpose 8x8	12	8 x 8	0.19	**12.5**
	1/2-pel interpolation 16x16	83	16 x 16	1.30	**9.8**
MPEG-4	Vertical lowpass filter	453	16 x 16	7.08	**12.9**
	Horizontal lowpass filter	516	16 x 16	8.06	**9.8**
H.264	Integer transform	82	4x4	5.13	**10.3**
	Inter-prediction				
	vertical filtering	71	8x8	1.11	**25.2**
	horizontal filtering	116	8x8	1.81	**15.4**
	1/4-pel interpolation	46	8x8	0.72	**8.3**
	Intra-prediction (planar)	240	16x16	0.94	**7.7**
	De-blocking filter				
	horizontal filtering	2	6	0.33	**70.0**
	vertical filtering	54	24	2.25	**10.4**
VC-1	Integer transform	143	8 x 8	2.23	**10.7**
	Bilinear block filter	93	8 x 8	1.45	**11.0**
	Bicubic 2D filter	216	8 x 8	3.38	**11.0**

SIMD-Based ARC™ Media Subsystem Implementation

- SIMD extensions add 150K gates
- Shares ARC™ 750D core's low-power characteristics
- Capable of >500 MHz operation
  - Codecs typically require 166 MHz or less
- Platform is complete with codec libraries
  - H.264 decode, baseline + main
  - VC-1 decode, baseline + main
  - MPEG-2, MP@ML
  - MPEG-4, SP to ASP L4
  - JPEG, M-JPEG encode
  - AAC, MP3 12-channel encode
- Linux or standalone operation
- ARCangel™ 4 development platform
  - Virtex-4™ FPGA runs codecs real-time

ARC™ Media Subsystem Device[1]

ARC 750D core with 32K I-cache, 32K D-cache
+ SIMD extensions
+ Entropy decode extensions
+ DMA
+ 32K SIMD data memory
+ 10K SIMD code memory

9mm²

Other multicore solutions[2]…

42mm²

1. Synthesized in 0.13µm technology for operation at 500 MHz
2. Scaled to 0.13µm for direct comparison, from 81mm² at 0.18µm, drawn approximately to scale

Summary

- SIMD extensions deliver high performance, yet low power
- Extends ARC™ 700 core family with 104 new instructions
- Decoupled programming model maximizes parallelism
  - Natural and simple extension to single-thread model
- Media-oriented DMA shoulders data transfers
- Supports programmable multistandard audio / video codecs
  - Competitive with dedicated hardware solutions
- Supported by full suite of ARC and GNU software tools
- Integrates with third-party IP
- ARC Media subsystem available Q4
- Symmetric multicore
  - ARC-based implementations now
  - New core products in 2006

ARC™ Media Subsystem Floorplan

ARC 750D core with 32K I-cache, 32K D-cache
+ SIMD extensions
+ Entropy decode (MPEG-2, MPEG-4, H.264, VC-1)
+ DMA
+ 32K SIMD data memory
+ 10K SIMD code memory
Area: 9mm² in 0.13µm TSMC, Virage libraries
Speed: 500 MHz (worst-case, LVLK-OD process)